APPLICATION FOR UNITED STATES PATENT

INVENTOR: Paul S. Neuman

INVENTION: METHOD FOR IMPROVED FIRST LEVEL CACHE COHERENCY

DOCKET NUMBER: RA 5290 (33012/289/101))

Unisys Corporation
Charles A. Johnson
P O Box 64942 - MS 4773
St. Paul, MN 55164
Attorney for Applicant
Reg. No.: 20,852

SPECIFICATION

<u>CERTIFICATE UNDER 37 CFR 1.10:</u> The undersigned hereby certifies that this transmittal letter and the paper of papers, as described hereinabove, are being deposited in the United States Postal Service, "Express Mail Post Office to Addressee" having an Express Mail mailing label number of EL 027383860 US, in an envelope addressed to: ASSISTANT COMMISSIONER OF PATENTS, Washington, D.C. 20231 on the 30th day of August, 2000.

[illegible]

Charles A. Johnson

METHOD FOR IMPROVED FIRST LEVEL CACHE COHERENCY

CROSS REFERENCE TO CO-PENDING APPLICATIONS

The present application is related to co-pending U.S. Patent Application Serial No. ____________, filed ______________, entitled Cooperative Hardware and Microcode Control System for Pipelined Instruction Execution; U.S. Patent Application Serial No. ____________, filed ______________, entitled Method for Avoiding Delays During SNOOP Requests; U.S. Patent Application Serial No. ____________, filed ______________, entitled Leaky Cache Mechanism; and U.S. Patent Application Serial No. ____________, filed ______________, entitled Data Coherency Protocol for Multi-level Cached High Performance Multiprocessor System, assigned to the assignee of the present invention and incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. **Field of the Invention:** - The present invention relates generally to data processing systems employing multiple instruction processors and more particularly relates to multiprocessor data processing systems employing multiple levels of cache memory.

2. **Description of the Prior Art:** - It is known in the art that the use of multiple instruction processors operating out of

common memory can produce problems associated with the processing of obsolete memory data by a first processor after that memory data has been updated by a second processor. The first attempts at solving this problem tended to use logic to lock processors out of memory spaces being updated. Though this is appropriate for rudimentary applications, as systems become more complex, the additional hardware and/or operating time required for the setting and releasing of locks can not be justified, except for security purposes. Furthermore, reliance on such locks directly prohibits certain types of applications such as parallel processing.

The use of hierarchical memory systems tends to further compound the problem of data obsolescence. U.S. Patent No. 4,056,844 issued to Izumi shows a rather early approach to a solution. The system of Izumi utilizes a buffer memory dedicated to each of the processors in the system. Each processor accesses a buffer address array to determine if a particular data element is present in its buffer memory. An additional bit is added to the buffer address array to indicate invalidity of the corresponding data stored in the buffer memory. A set invalidity bit indicates that the main storage has been altered at that location since loading of the buffer memory. The validity bits are set in accordance with the memory store cycle of each processor.

12/10/02
PNV

U.S. Patent No. 4,349,871 issued to Lary describes a bussed architecture having multiple processing elements, each having a dedicated cache memory. According to the Lary design, each

12/10/02
PNV

processing unit manages its own cache by monitoring the memory bus. Any invalidation of locally stored data is tagged to prevent use of obsolete data. The overhead associated with this approach is partially mitigated by the use of special purpose hardware and through interleaving the validity determination with memory accesses within the pipeline. Interleaving of invalidity determination is also employed in U.S. Patent No. 4,525,777 issued to Webster et al.

Similar bussed approaches are shown in U.S. Patent No. 4,843,542 issued to Dashiell et al, and in U.S. Patent No. 4,755,930 issued to Wilson, Jr. et al. In employing each of these techniques, the individual processor has primary responsibility for monitoring the memory bus to maintain currency of its own cache data. U.S. Patent No. 4,860,192 issued to Sachs et al, also employs a bussed architecture but partitions the local cache memory into instruction and operand modules.

12/10/02
PMV

U.S. Patent No. 5,025,365 issued to Mathur et al, provides a much enhanced architecture for the basic bussed approach. In Mathur et al, as with the other bussed systems, each processing element has a dedicated cache resource. Similarly, the cache resource is responsible for monitoring the system bus for any collateral memory accesses which would invalidate local data. Mathur et al, provide a special snooping protocol which improves system throughput by updating local directories at times not necessarily coincident with cache accesses. Coherency is assured

12/10/02
PMV

by the timing and protocol of the bus in conjunction with timing of the operation of the processing element.

An approach to the design of an integrated cache chip is shown in U.S. Patent No. 5,025,366 issued to Baror. This device provides the cache memory and the control circuitry in a single package. The technique lends itself primarily to bussed architectures. U.S. Patent No. 4,794,521 issued to Ziegler et al, shows a similar approach on a larger scale. The Ziegler et al, design permits an individual cache to interleave requests from multiple processors. This design resolves the data obsolescence issue by not dedicating cache memory to individual processors. Unfortunately, this provides a performance penalty in many applications because it tends to produce queuing of requests at a given cache module.

12/10/02
pnv

The use of a hierarchical memory system in a multiprocessor environment is also shown in U.S. Patent No. 4,442,487 issued to Fletcher et al. In this approach, each processor has dedicated and shared caches at both the L1 or level closest to the processor and at the L2 or intermediate level. Memory is managed by permitting more than one processor to operate upon a single data block only when that data block is placed in shared cache. Data blocks in dedicated or private cache are essentially locked out until placed within a shared memory element. System level memory management is accomplished by a storage control element through which all requests to shared main memory (i.e. L3 level) are routed. An apparent improvement to this approach is shown in U.S. Patent No.

12/10/02
pnv

4,807,110 issued to Pomerene et al. This improvement provides prefetching of data through the use of a shadow directory.

12/10/02
pmv

A further improvement to Fletcher et al, is seen in U.S. Patent No. 5,023,776 issued to Gregor. In this system, performance can be enhanced through the use of store around L1 caches used along with special write buffers at the L2 intermediate level. This approach appears to require substantial additional hardware and entails yet more functions for the system storage controller.

SUMMARY OF THE INVENTION

The present invention overcomes the problems found in the prior art by providing a method and apparatus for improving the efficiency of maintaining coherency of a first level cache memory within a system having multiple levels of cache memory. This enhancement to efficiency is accomplished by utilizing a validation cycle, invalidation cycle, and parity detection circuitry, as discussed below.

The preferred mode of the present invention includes up to four main memory storage units. Each is coupled directly to each of up to four "pod"s. Each pod contains a level three cache memory coupled to each of the main memory storage units. Each pod may also accommodate up to two input/output modules.

Each pod may contain up to two sub-pods, wherein each sub-pod may contain up to two instruction processors. Each instruction processor has two separate level one cache memories (one for instructions and one for operands) coupled through a dedicated system controller, having a second level cache memory, to the level three cache memory of the pod.

Unlike many prior art systems, both level one and level two cache memories are dedicated to an instruction processor within the preferred mode of the present invention. The level one cache memories are of two types. Each instruction processor has an instruction cache memory and an operand cache memory. The instruction cache memory is a read-only cache memory primarily having sequential access. The level one operand cache memory has read/write capability. In the read mode, it functions much as the level one instruction cache memory. In the write mode, it is a semi-store-in cache memory, because the level two cache memory is also dedicated to the instruction processor.

In accordance with the present invention, level one cache memory data is invalidated during the invalidate cycle under three conditions. In the third mode, a write hit causes invalidation as a result of declared ownership of the data. In the first or second modes, write hits or system bus snoop hits result in invalidation of the corresponding level one data.

When read data is made available, the system controller maintains a record of the location of the data within the level one cache memory.

Finally, a parity error within the system controller level two cache memory causes invalidation of the level one cache memory data. This is done as a precaution against loss of coherency control.

BRIEF DESCRIPTION OF THE DRAWINGS

Other objects of the present invention and many of the attendant advantages of the present invention will be readily appreciated as the same becomes better understood by reference to the following detailed description when considered in connection with the accompanying drawings, in which like reference numerals designate like parts throughout the figures thereof and wherein:

FIG. 1 is an overall block diagram of a fully populated system in accordance with the present invention;

FIG. 2 is a schematic block diagram of one pod;

FIG. 3 is a schematic block diagram of one instruction processor along with its dedicated system controller; and

FIG. 4 is a detailed flow chart of the operation of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

FIG. 1 is an overall block diagram of fully populated data processing system according to the preferred mode of the present invention. This corresponds to the architecture of a commercial system of Unisys Corporation termed "Voyager".

The main memory of the system consists of up to four memory storage units, MSU 10, MSU 12, MSU 14, and MSU 16. Being fully modular, each of these four memory storage units is "stand-alone" and independent of one another. Each has a separate point-to-point dedicated bi-directional interface with up to four "pods", POD 18, POD 20, POD 22, POD 24. Again, each of the up to four pods is separate and independent of one another.

The contents of POD 20 are shown by way of example. For the fully populated system, POD 18, POD 22, and POD 24 are identical to POD 20. The interface between POD 20 and each of the four memory storage units (i.e., MSU 10, MSU 12, MSU 14, and MSU 16), is via a third level cache memory designated cached interface, CI 26, in this view. CI 26 couples with two input/output controllers, I/O Module 44 and I/O Module 46, and two sub-pods, SUB 28 and SUB 30. A more detailed explanation of the POD 20 is provided below.

The above described components are the major data handling elements of the system. In the fully populated system shown, there are sufficient components of each type, such that no single hardware failure will render the complete system inoperative. The software employed within the preferred mode of the present system utilizes these multiple components to provide enhanced reliability for long term operation.

The remaining system components are utilitarian rather than data handling. System Oscillator 32 is the primary system time and clocking standard. Management System 34 controls system testing, maintenance, and configuration. Power Controller 36 provides the required electrical power. System Oscillator 38, Management System 40, and Power Controller 42 provide completely redundant backup capability.

FIG. 2 is a more detailed block diagram of POD 20. The level three cache memory interfaces directly with the memory storage units via TLC Controller 26 (see also Fig. 1). The actual storage for the level three cache memory is TLC SRAMS 48. As indicated this static random access memory consists of eight 16 byte memory chips.

Subpod 28 and subpod 30 each contain up to two individual instruction processors. These are designated Voyager IP 50, Voyager IP 52, Voyager IP 54, and Voyager IP 56. As explained in detail below, each contains its own system controller. In accordance with the preferred mode of the present invention, these instruction processors need not all contain an identical software architecture.

FIG. 3 is a more detailed block diagram of Voyager IP 50, located within Subpod 28, located within POD 20 (see also Figs. 1 and 2). As explained above, each instruction processor has a dedicated system controller having a dedicated level two cache memory. Instruction processor 64 has two dedicated level one cache memories (not shown in this view). One level one cache memory is a read-only memory for program instruction storage. Instruction processor 64 executes its instructions from this level one cache memory. The other level one cache memory (also not shown in this view) is a read/write memory for operand storage.

Instruction processor 64 is coupled via its two level one cache memories and dedicated system controller 58 to the remainder of the system. System controller 58 contains input logic 74 to interface with instruction processor 64. In addition, data path logic 70 controls movement of the data through system controller 58. The utilitarian functions are provided by Locks, Dayclocks, and UPI 62.

The remaining elements of system controller 58 provide the level two cache memory functions. SLC data ram 66 is the data actual storage facility. Control logic 70 provides the cache management function. SLC tags 72 are the tags associated with the level two cache memory. FLC-IC Dup. Tags 76 provides the duplicate tags for the level one instruction cache memory of instruction processor 64. Similarly, FLC-OC Dup. Tags 78 provides the duplicate tags for the level one operand cache memory of instruction processor 64. For a more complete discusses of this duplicate tag approach, reference may be made with the above identified co-pending and incorporated U.S. Patent Applications.

FIG. 4 is a detailed flow chart of the operation of the present invention showing three specific opportunities to improve efficiency of the multi-level cache memory system. Process 80 provides for invalidation of level one cache memory contents based upon conditions during the invalidate cycle. Process 82 is an enhancement based upon the validation cycle. Process 84 provides invalidation upon a parity error.

Process 80 is performed as a result of an instruction processor write request (i.e., an operand write) or memory bus SNOOP (i.e., monitoring of memory busses involving other instruction processors and other system controllers). For either of these activities, the corresponding data within the level one operand cache memory is invalidated under the shown circumstances. This invalidation during the invalidation cycle provides a time saving over a subsequent potential future request for the invalidated data.

Element 86 determines whether a write request has been made of the system controller (see also Fig. 3). Again, this request means that the instruction processor has attempted an operand write. If yes, element 88 determines whether this is a mode_3. If yes, element 92 requires ownership of the data within the level one operand cache memory or control is returned to element 94 without action. Note that a mode 3 write without ownership means that there is no data within the level one operand cache memory to invalidate. Therefore, further activity within process 80 would be wasted.

However, if element 92 determines that ownership is present, control is given to element 90 which determines whether there is a level one cache memory hit (i.e., operand write is to a data element present within the level one cache memory). If there is a hit, element 102 invalidates the contents of the corresponding location with the level one operand cache memory. However, if there is no hit, there can be no time saving, because a memory operation involving the level three cache memory is required.

The other condition for which process 80 is applicable is a system memory bus SNOOP. In this activity, the system controller monitors the system memory buses for activity from other processors and system controllers which might impact the validity of its local data. The above identified commonly assigned, co-pending, and incorporated patent applications discuss the SNOOP activity in greater detail.

The SNOOP activity is sensed by element 94. When present, after checking for Mode 3, control is given to element 96 to determine whether it is mode 1. Element 100 determines whether the SNOOP experiences a hit on the bus read line (BRL). If no, control is returned to element 104, because modes 2 and 3 are not concerned with the bus read line. Element 98 determines whether there is a hit on the bus write line (BWL) or bus read invalidate line (BRIL).

If there is a hit, element 102 invalidates the corresponding location within the level one cache memory. This improves efficiency, because the data is invalidated without even being requested.

Read access is the concern of process 82. Element 104 determines when a read occurs. This is a result of an operand read operation within the instruction processor which experiences a level one operand cache memory miss. The request is made to the system controller wherein element 104 gives control to element 106. If there is also a miss within the level two cache memory, control is given to element 108 for retrieval of the requested data from the level three cache memory. After receiving the requested data, element 108 also loads the requested data into the level two cache memory and records its presence and location within the level one cache memory, thus making it potentially available for in response to a subsequent request.

The purpose of process 84 is an error recovery technique. Whenever element 112 determines that a parity error has been experienced with regard to the level two cache memory. When a parity error occurs, element 110 causes invalidation of the corresponding locations within the level one cache memory, to avoid loss of control between the instruction processor level one cache memories and the system controller level two cache memory.

Having thus described the preferred embodiments in sufficient detail for those of skill in the art to make and use the present invention, those of skill in the art will be readily able to apply the teachings found herein to yet other embodiments within the scope of the claims hereto attached.

WE CLAIM: